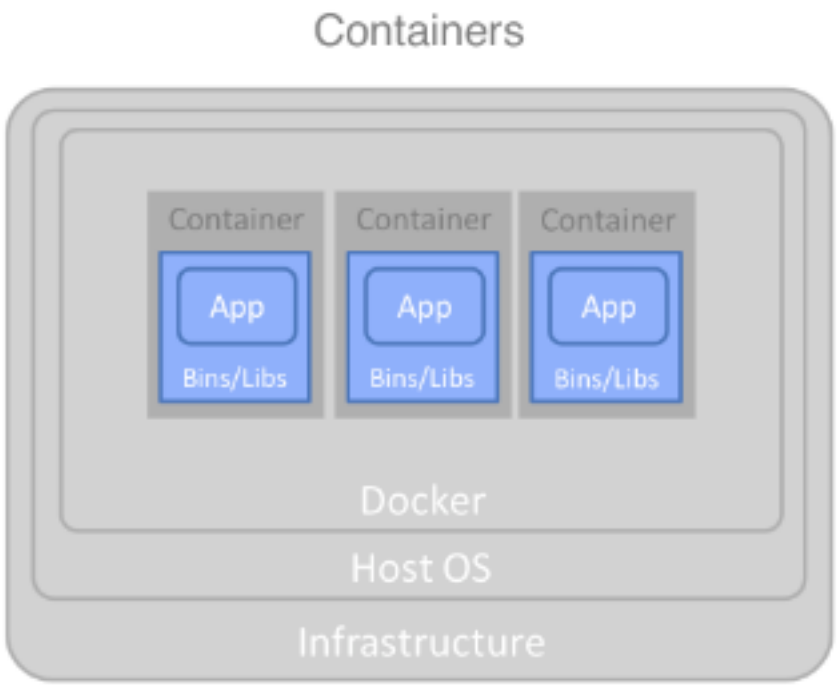
Exhibit 4

U.S. Patent No. 7,784,058 ("'058 Patent")

Accused Instrumentalities: IBM's IBM Cloud Kubernetes Service, and all versions and variations thereof since the issuance of the asserted patent.

Claim 1

Claim 1	Accused Instrumentalities
[1pre] 1. A computing system for executing a plurality of software applications comprising:	To the extent the preamble is limiting, each Accused Instrumentality comprises or constitutes a computing system for executing a plurality of software applications as claimed. *See* claim limitations below. *See also, e.g.*: IBM Cloud® Kubernetes Service provides a fully managed container service for Docker (OCI) containers, so clients can deploy containerized apps onto a pool of compute hosts and subsequently manage those containers. Containers are automatically scheduled and placed onto available compute hosts based on your requirements and availability in the cluster. https://www.ibm.com/products/kubernetes-service With IBM Cloud Kubernetes Service, you can deploy Docker containers into pods that run on your worker nodes. The worker nodes come with a set of add-on pods to help you manage your containers. Install more add-ons through Helm, a Kubernetes package manager. These add-ons can extend your apps with dashboards, logging, IBM Cloud and IBM Watson® services and more. https://www.ibm.com/products/kubernetes-service

Claim 1	Accused Instrumentalities
	 https://developer.ibm.com/articles/true-benefits-of-moving-to-containers-1/
[1a] a) a processor;	Each Accused Instrumentality comprises a processor. *See, e.g.*:

Claim 1	Accused Instrumentalities
	Containers are executable units of software in which application code is packaged along with its libraries and dependencies, in common ways so that the code can be run anywhere—whether it be on desktop, traditional IT or the cloud. To do this, containers take advantage of a form of operating system (OS) virtualization in which features of the OS kernel (e.g. Linux namespaces and cgroups, Windows silos and job objects) can be leveraged to isolate processes and control the amount of CPU, memory and disk that those processes can access. Containers are small, fast and portable because unlike a virtual machine, containers do not need to include a guest OS in every instance and can instead simply leverage the features and resources of the host OS. https://www.ibm.com/topics/containers Containers use a form of operating system (OS) virtualization. Put simply, they leverage features of the host operating system to isolate processes and control the processes' access to CPUs, memory and desk space. https://www.ibm.com/blog/containers-vs-vms/

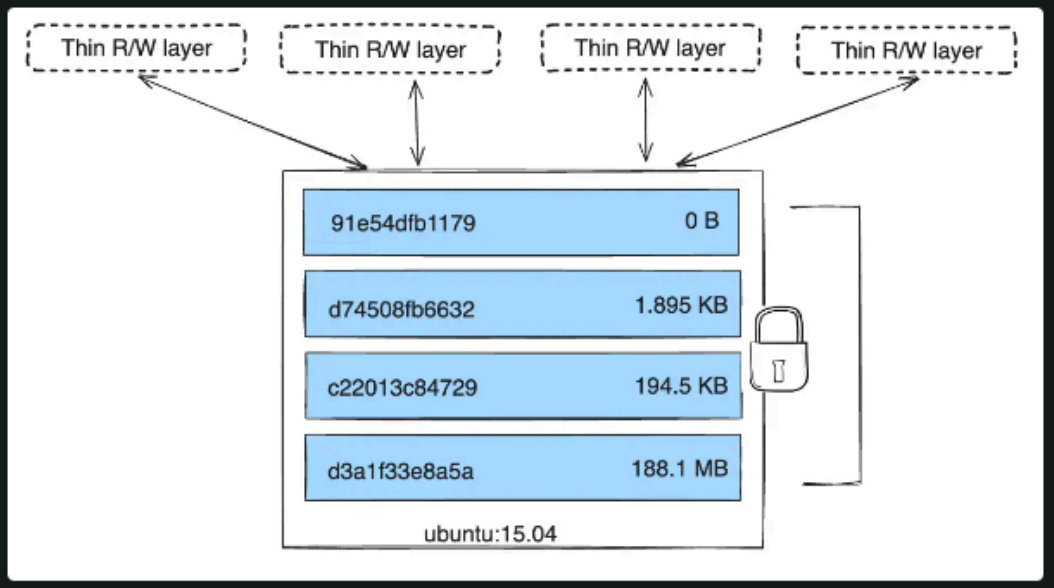
Claim 1	Accused Instrumentalities
[1b] b) an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor; and,	Each Accused Instrumentality comprises an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor. *See, e.g.*: Containerization is the packaging of software code with just the operating system (OS) libraries and dependencies required to run the code to create a single lightweight executable—called a container—that runs consistently on any infrastructure. More portable and resource-efficient than virtual machines (VMs), containers have become the de facto compute units of modern cloud-native applications. Containerization allows developers to create and deploy applications faster and more securely. With traditional methods, code is developed in a specific computing environment which, when transferred to a new location, often results in bugs and errors. For example, when a developer transfers code from a desktop computer to a VM or from a Linux to a Windows operating system. Containerization eliminates this problem by bundling the application code together with the related configuration files, libraries, and dependencies required for it to run. This single package of software or "container" is abstracted away from the host operating system, and hence, it stands alone and becomes portable—able to run across any platform or cloud, free of issues. https://www.ibm.com/topics/containerization **Kernel mode** Kernel mode refers to the processor mode that enables software to have full and unrestricted access to the system and its resources. The OS kernel and kernel drivers, such as the file system driver, are loaded into protected memory space and operate in this highly privileged kernel mode. https://www.techtarget.com/searchdatacenter/definition/kernel

Claim 1	Accused Instrumentalities
[1c] c) a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode and	Each Accused Instrumentality comprises a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode. *See, e.g.*: A Docker image is the basis for every container that you create with IBM Cloud® Kubernetes Service. An image is created from a Dockerfile, which is a file that contains instructions to build the image. A Dockerfile might reference build artifacts in its instructions that are stored separately, such as an app, the app's configuration, and its dependencies. https://cloud.ibm.com/docs/containers?topic=containers-images

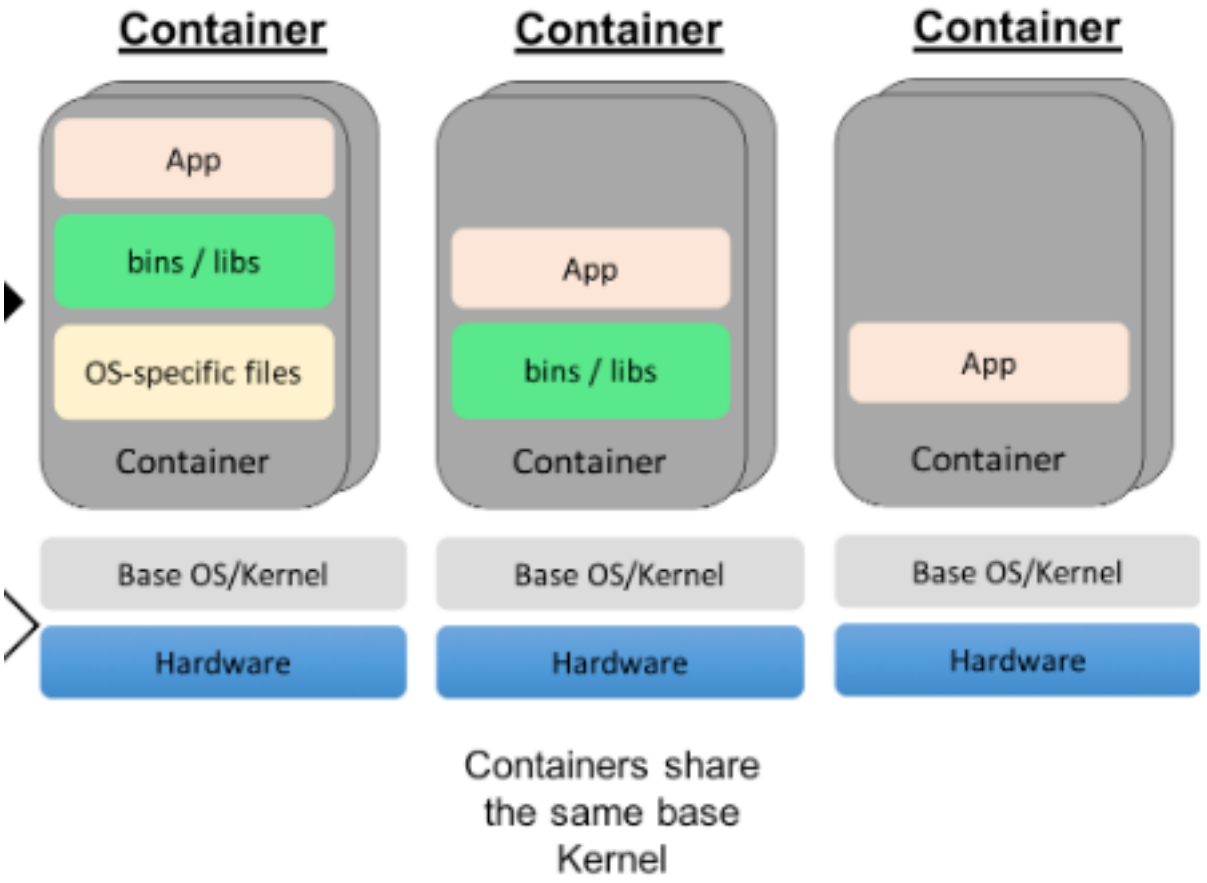
Claim 1	Accused Instrumentalities

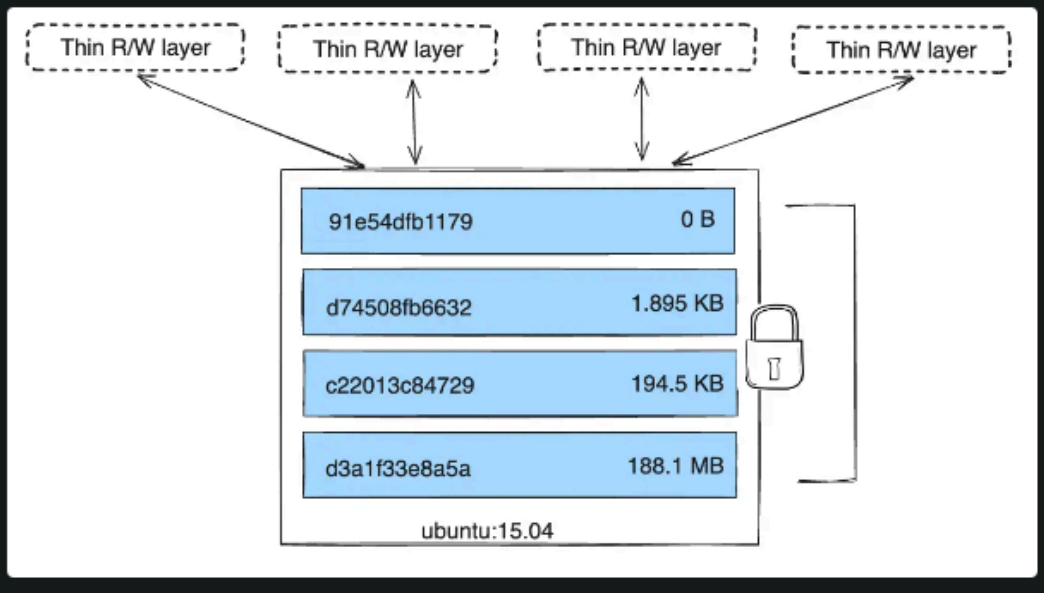
Docker base image	Supported versions	Source of security notices
Alpine	All stable versions with vendor security support.	Alpine SecDB database .
Debian	All stable versions with vendor security support. CVEs on binary packages that are associated with the Debian source package `linux`, such as `linux-libc-dev`, are not reported. Most of these binary packages are kernel and kernel modules, which are not run in container images.	Debian Security Bug Tracker .
GoogleContainerTools distroless	All stable versions with vendor security support.	GoogleContainerTools distroless
Red Hat® Enterprise Linux® (RHEL)	RHEL/UBI 7, RHEL/UBI 8, and RHEL/UBI 9	Red Hat Security Data API .
Ubuntu	All stable versions with vendor security support.	Ubuntu CVE Tracker .

https://cloud.ibm.com/docs/Registry?topic=Registry-va_index&interface=ui

Claim 1	Accused Instrumentalities
	Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image.  https://docs.docker.com/storage/storagedriver/

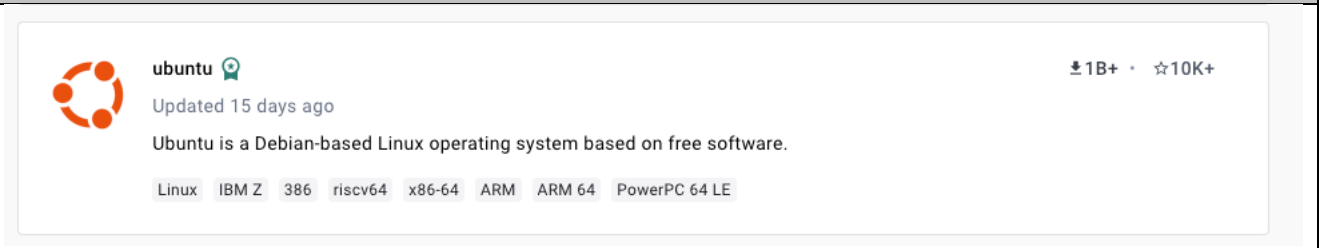
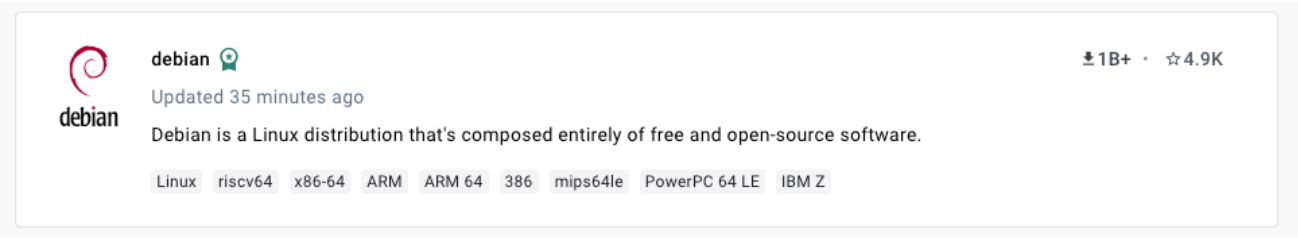
Claim 1	Accused Instrumentalities
	Docker images contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes one instance (or multiple instances) of the container. It's possible to build a Docker image from scratch, but most developers pull them down from common repositories. Multiple Docker images can be created from a single base image, and they'll share the commonalities of their stack. Docker images are made up of layers, and each layer corresponds to a version of the image. Whenever a developer makes changes to the image, a new top layer is created, and this top layer replaces the previous top layer as the current version of the image. Previous layers are saved for rollbacks or to be re-used in other projects. Each time a container is created from a Docker image, yet another new layer called the container layer is created. Changes made to the container—such as the addition or deletion of files—are saved to the container layer only and exist only while the container is running. This iterative image-creation process enables increased overall efficiency since multiple live container instances can run from just a single base image, and when they do so, they leverage a common stack. https://www.ibm.com/topics/docker

Claim 1	Accused Instrumentalities
	 https://ibm.github.io/kube101/

Claim 1	Accused Instrumentalities
	Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image.  https://docs.docker.com/storage/storagedriver/
[1d] i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of	In each Accused Instrumentality, some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications.

Claim 1	Accused Instrumentalities
software applications and when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications,	For example, a Docker base image serves as a self-contained unit that encompasses all the necessary components for an application to run, including the application code, runtime environment, system tools, and dependencies (i.e., SLCSEs). The images are based on existing Linux distributions, such as Debian and Ubuntu, including essential system elements (i.e., functional replicas of OSCSEs). Each container image is based on a specific base image, which contains the application code, and dependencies, including system libraries or shared library critical system elements (SLCSEs). When the container runs the image, it creates a runtime instance of that container image. *See, e.g.*: A Docker image is the basis for every container that you create with IBM Cloud® Kubernetes Service. An image is created from a Dockerfile, which is a file that contains instructions to build the image. A Dockerfile might reference build artifacts in its instructions that are stored separately, such as an app, the app's configuration, and its dependencies. https://cloud.ibm.com/docs/containers?topic=containers-images

Claim 1	Accused Instrumentalities
	Docker images contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes one instance (or multiple instances) of the container. It's possible to build a Docker image from scratch, but most developers pull them down from common repositories. Multiple Docker images can be created from a single base image, and they'll share the commonalities of their stack. Docker images are made up of layers, and each layer corresponds to a version of the image. Whenever a developer makes changes to the image, a new top layer is created, and this top layer replaces the previous top layer as the current version of the image. Previous layers are saved for rollbacks or to be re-used in other projects. Each time a container is created from a Docker image, yet another new layer called the container layer is created. Changes made to the container—such as the addition or deletion of files—are saved to the container layer only and exist only while the container is running. This iterative image-creation process enables increased overall efficiency since multiple live container instances can run from just a single base image, and when they do so, they leverage a common stack. https://www.ibm.com/topics/docker Following software is installed on the Docker containers as part of the Product Master image deployment: – Red Hat Enterprise Linux (RHEL) 7 Universal Base Image (UBI) base Docker image https://www.ibm.com/docs/en/product-master/12.0.0?topic=deployment-installing-product-by-using-docker-images

Claim 1	Accused Instrumentalities
	  https://hub.docker.com/search?image_filter=official&type=image&q=

Claim 1	Accused Instrumentalities

Platform	x86_64 / amd64	arm64 / aarch64	arm (32-bit)	ppc64le	s390x
CentOS	✅	✅		✅	
Debian	✅	✅	✅	✅	
Fedora	✅	✅		✅	
Raspberry Pi OS (32-bit)			✅		
RHEL (s390x)					✅
SLES					✅
Ubuntu	✅	✅	✅	✅	✅
Binaries	✅	✅	✅		

https://docs.docker.com/engine/install/

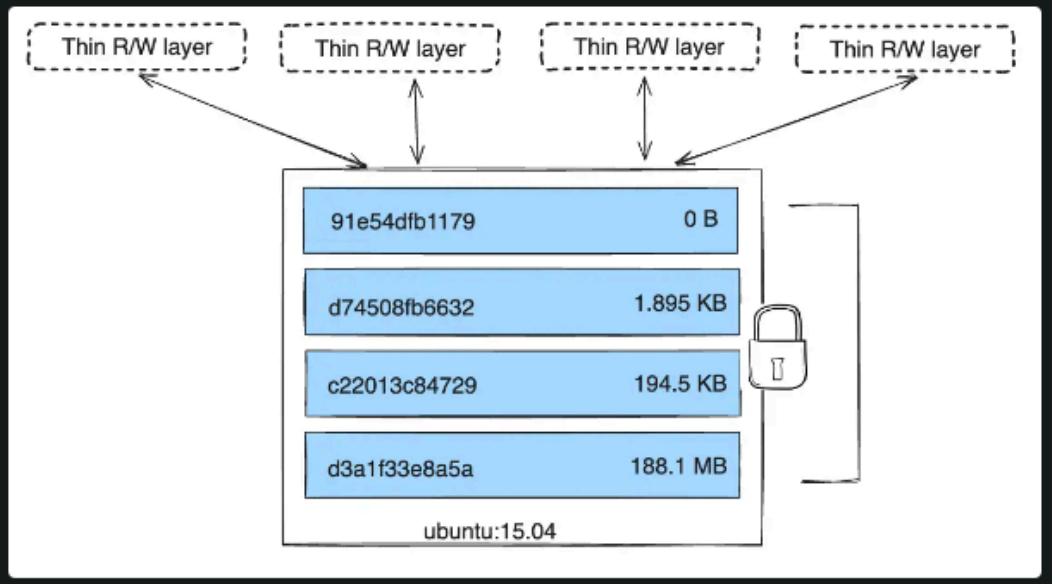
Docker is used to create, run and deploy applications in containers. A Docker image contains application code, libraries, tools, dependencies and other files needed to make an application run. When a user runs an image, it can become one or many instances of a container.

https://www.techtarget.com/searchitoperations/definition/Docker-image

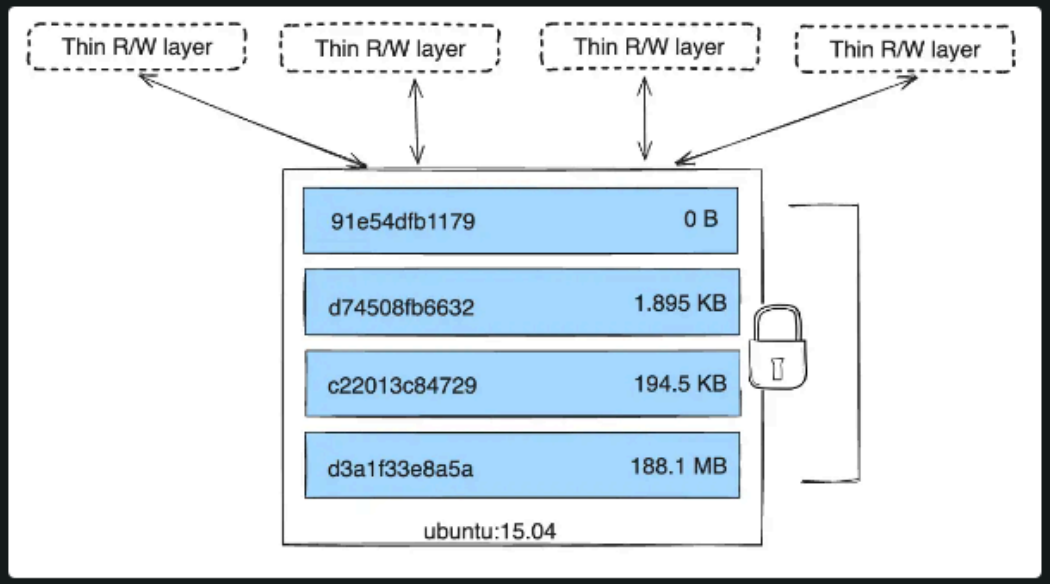
Docker images contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes one instance (or multiple instances) of the container.

https://www.ibm.com/topics/docker

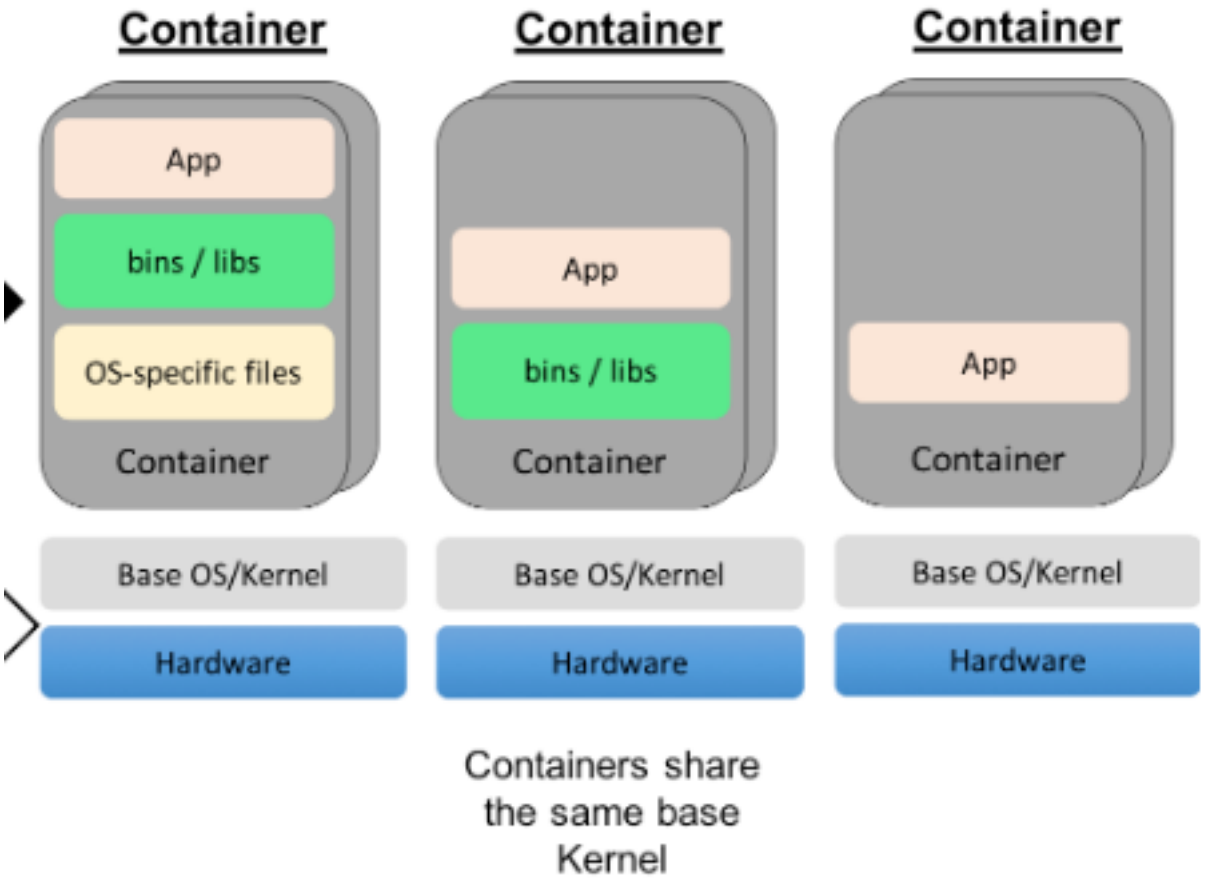
Claim 1	Accused Instrumentalities
	container A container is a runtime instance of a docker image. A Docker container consists of • A Docker image • An execution environment • A standard set of instructions https://docs.docker.com/glossary/#image

Claim 1	Accused Instrumentalities
	Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image.  https://docs.docker.com/storage/storagedriver/
[1e] ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software	In each Accused Instrumentality, an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function.

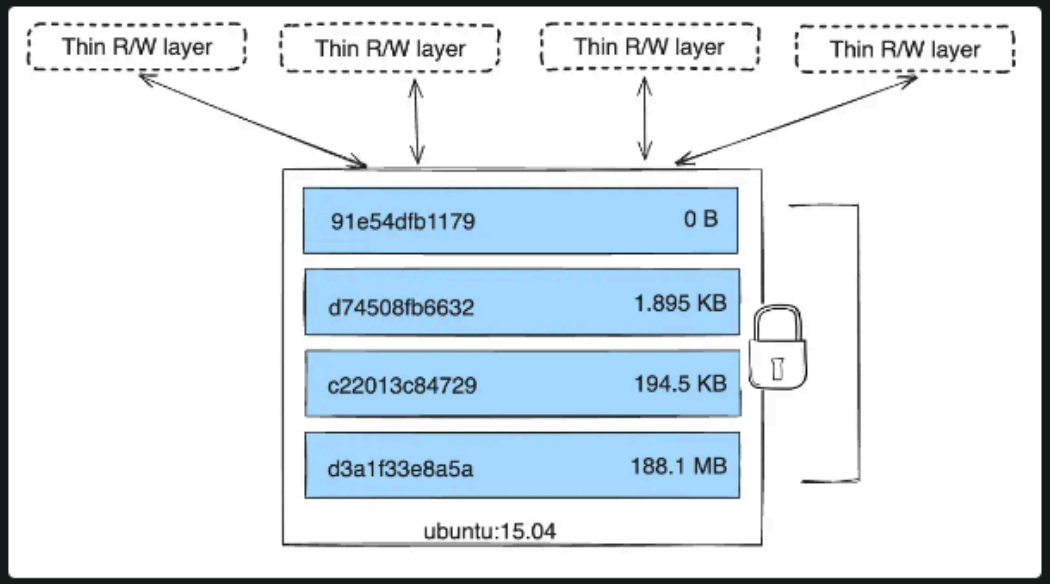
Claim 1	Accused Instrumentalities
applications without being shared with other of the plurality of software applications and where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and	When a Docker image is used to create a container, it creates a separate and isolated instance of a runtime environment which is independent of other containers running on the same host. Each container has its own instance of base images and its own data. The containers run in isolation, ensuring that the SLCSEs stored in the shared library are accessible to the software applications running in their respective containers. The docker image includes essential system files, libraries, and dependencies required to run the software application within the container. The Docker containers can share common dependencies and components using layered images. This means that multiple containers utilize the same base image to create an instance. When an instance of SLCSE is provided from the base image (i.e., from the shared library) to an individual container including application software, it operates in isolation and runs its own instance of the software application without sharing resources or critical system elements with other containers. This ensures that each container has its own isolated context. Docker containers can share common dependencies and components using layered images. This means that multiple containers can utilize the same base image. Therefore, each container, containing the application software running under the operating system, utilizes a unique instance of the corresponding critical system element to execute the respective application software for performing a same or a different function. *See, e.g.*: *Docker images* contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes one instance (or multiple instances) of the container. It's possible to build a Docker image from scratch, but most developers pull them down from common repositories. Multiple Docker images can be created from a single base image, and they'll share the commonalities of their stack. https://www.ibm.com/topics/docker

Claim 1	Accused Instrumentalities
	Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image.  https://docs.docker.com/storage/storagedriver/

Claim 1	Accused Instrumentalities
	Docker images contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes one instance (or multiple instances) of the container. https://www.ibm.com/topics/docker Docker is used to create, run and deploy applications in containers. A Docker image contains application code, libraries, tools, dependencies and other files needed to make an application run. When a user runs an image, it can become one or many instances of a container. https://www.techtarget.com/searchitoperations/definition/Docker-image

Claim 1	Accused Instrumentalities
	 https://ibm.github.io/kube101/

Claim 1	Accused Instrumentalities
[1f] iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.	In each Accused Instrumentality, a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously. For example, In Docker, each container operates independently, and a Docker base image includes essential system files, libraries, and dependencies (i.e., SLCSEs) required to run the software application within the container. Based on information and belief, each element, such as system files, libraries, and dependencies (i.e., SLCSE) is associated with an execution of a predetermined function related to the application. When a Docker image is used to create a container in ECS, an instance of the SLCSE is provided to a software application. Therefore, different instances of the SLCSE are provided to different applications for performing either a same or a different function, simultaneously. *See, e.g.*: Docker is used to create, run and deploy applications in containers. A Docker image contains application code, libraries, tools, dependencies and other files needed to make an application run. When a user runs an image, it can become one or many instances of a container. https://www.techtarget.com/searchitoperations/definition/Docker-image *Docker images* contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes one instance (or multiple instances) of the container. https://www.ibm.com/topics/docker

Claim 1	Accused Instrumentalities
	Because each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state. The diagram below shows multiple containers sharing the same Ubuntu 15.04 image.  https://docs.docker.com/storage/storagedriver/ A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

Claim 1	Accused Instrumentalities
	https://docs.docker.com/get-started/overview/